Benchmarking Faceted Browsing Capabilities of Triplestores

Henning Petzka
Fraunhofer Institute IAIS
Schloss Birlinghoven
53757 Sankt Augustin, Germany
University of Bonn
Roemerstrasse 164
53117 Bonn, Germany
henning.petzka@iais.fraunhofer.de

Claus Stadler
University of Leipzig
Augustusplatz 10
04103 Leipzig, Germany
cstadler@informatik.uni-leipzig.de

Georgios Katsimpras
NCSR "Demokritos"
Neapoleos
Ag. Paraskevi 153 10, Greece
gkatsibras@iit.demokritos.gr

Bastian Haarmann
Fraunhofer Institute IAIS
Schloss Birlinghoven
53757 Sankt Augustin, Germany
bastian.haarmann@iais.fraunhofer.de

Jens Lehmann
Fraunhofer Institute IAIS
Schloss Birlinghoven, 53757 Sankt Augustin
University of Bonn
Roemerstrasse 164, 53117 Bonn
jens.lehmann@cs.uni-bonn.de

ABSTRACT

The increasing availability of large amounts of linked data creates a need for software that allows for its efficient exploration. Systems enabling faceted browsing constitute a user-friendly solution that need to combine suitable choices for front and back end. Since a generic solution must be adjustable with respect to the dataset, the underlying ontology and the knowledge graph characteristics raise several challenges and heavily influence the browsing experience. As a consequence, an understanding of these challenges becomes an important matter of study. We present a benchmark on faceted browsing of triple stores, which allows systems to test their performance on specific choke points on the back end. Further, we address additional issues in faceted browsing that may be caused by problematic modelling choices within the underlying ontology.

CCS CONCEPTS

• **Information systems** → *Database performance evaluation*;

KEYWORDS

Faceted Browsing, Benchmarking, Choke Points, Linked Data

ACM Reference format:
Henning Petzka, Claus Stadler, Georgios Katsimpras, Bastian Haarmann, and Jens Lehmann. 2017. Benchmarking Faceted Browsing Capabilities of Triplestores. In *Proceedings of Semantics2017, Amsterdam, Netherlands, September 11–14, 2017,* 8 pages.
https://doi.org/10.1145/3132218.3132242

Semantics2017, September 11–14, 2017, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5296-3/17/09…$15.00
https://doi.org/10.1145/3132218.3132242

1 INTRODUCTION

Faceted Browsing stands for a session-based and state-dependent interactive method for query formulation over a multi-dimensional information space. It provides a user with an effective way for exploration of a search space. In the setting of linked data, where the dataset consists of a knowledge graph defined by triples of subject, predicate and object, faceted browsing is a particularly suitable method for exploration as the graph structure can be used to suggest promising edge-based transitions to the user. After a definition of the initial search space, i.e., the set of resources of interest to the user, a browsing scenario consists of applying (or removing) filter restrictions defined by object-valued properties or of changing the range of a property value of various data types. Those properties that can be used from a certain state in a browsing scenario can be easily and automatically read off from the knowledge graph.

For a well-established example of an implementation of faceted browsing consider an online shopping portal. There, the search space could be a certain type of clothes and, amongst others, the facets could be the size, color and price of clothes. Using filtering operations, the user is able to browse between different states of the search space in order to select the items with the desired properties.

As a second example, consider logs of a machine in an industrial setting. An engineer might want to browse through the log data of the machines to search for malfunctions. He or she may select and unselect specific machines, specific variables, and narrow down the data selection by specifying a range of measurement results (for example to find incidences involving very high temperatures). The increasing awareness of the value of industrial data and its increasing collection create a need for software that enables faceted exploration and is at the same time easy to deploy on any given dataset.

We assume that the dataset to be explored is available in RDF/S format[1]. A browsing scenario is then characterized by its sequence

[1] https://www.w3.org/TR/rdf-schema/

of queries, which are written in the SPARQL query language[2]. Each query defines a transition from one state of the browsing scenario to the next one. Here, a state consists of the chosen facets, their corresponding facet values and the set of instances satisfying all chosen constraints.

In a browsing scenario it is then the effective transition from one state to the next one that determines the user experience. Ideally, a system uses information about the state of the browsing scenario to efficiently return its answer to the SPARQL query that makes up the desired transition, instead of answering the query on the basis of the entire dataset in its original form. An efficient system for faceted browsing supports these transitions where, quite possibly, choices have to be made whether to better support a certain transition or the other. Runtime performance improvements can be achieved through an intelligent database structure or through certain precomputations.

To support a user-friendly and efficient browsing experience, the system also has to be able to quickly compute the underlying statistics. The statistics can then help the graphical user interface to decide which of the facets should be displayed. In particular, no facet should be suggested to the user, which leads to an empty search space. Further, in an ideal setting, for each facet and facet-value, the number of instances that remain after applying this facet should be displayed to the user before its selection. This leads to so-called facet counts, which are the second element of a faceted browsing implementation.

The goal of a benchmark for faceted browsing techniques and tools is to equip solution providers with a way to check their software for their efficiency in navigating through large-scale structured datasets, where the navigation is driven by intelligent iterative restrictions. We present a benchmark, which was built driven by the goal to provide a platform for benchmarking faceted browsing systems on browsing scenarios through a dataset of triples in the RDF format, which reflect an authentic use-case and challenge participating systems on different points of difficulty.

While exploring industrial datasets that could be used within our benchmark, we came across additional problems for developers trying to build a generic system for faceted browsing. Certain choices in the modeling of the ontology seem to prevent generic faceted browsing solutions by requiring human-driven preprocessing steps. In other cases, a semantic understanding of properties and classes seems to be required to only display sensible transitions from a given state. We list these observations as additional choke points of faceted browsing, with the hope to prevent such modeling choices or to inspire other researchers for ideas how to overcome these difficulties. Im summary, our contributions are the following:

- We collected a detailed list of transition centered choke points for faceted browsing.
- We wrote SPARQL queries simulating real-world browsing scenarios and representing all of the collected choke points.
- We present a benchmark on faceted browsing based on our browsing scenarios. To our knowledge, this is the first benchmark that tests the performance on transition centered choke points.
- We present preliminary results from our benchmark.
- We discuss problems in developing generic and dataset independent faceted browsing solutions.

[2]https://www.w3.org/TR/rdf-sparql-query/

2 RELATED WORK

Traditionally, information search falls into the two categories of lookup (a.k.a. focalized) and exploratory search [16] [3]. In general, the former is concerned with, given a carefully specified query, how to obtain a result set with minimal need of examination. Thus, web search engines are the most prominent representatives. In contrast, defining characteristics of exploratory search includes imprecise task requirements or open-ended search goals. Faceted search is a widely used representative of the latter type.

A comprehensive survey on faceted search on RDF/S datasets has recently been performed by [25]. There, the authors analyze 30 systems in regard to a formal state-based model that captures the essential functionality of faceted exploration. On that basis, different types of transitions are identified, relating to classes, properties, property paths, and complex transition markers (used e.g. to handle blank nodes and support disjunctions). Further, the authors present a terminology alignment of common fundamental concepts in faceted search that in prior literature occurred under varying names.

Faceted exploration overlaps with OLAP (OnLine Analytical Processing) as both domains deal with multi-dimensional data, as each facet can be seen as a dimension. However, the main distinguishing element is, that OLAP cubes have both a fixed schema and information demand (such as sales per month per department) [4]. Under this perspective, it is unsurprising that faceted search systems have also been employed as a means to select resources of interest carrying statistical information (such as using the DataCube vocabulary) for subsequent visualization [13] [17].

While several publications emphasize on (the evaluation of) features implemented in a specific tool's user interface, a potential step towards increased reusability of solutions have been taken with the introduction of the *Query-based Faceted Search* paradigm [8], where aspects of faceted search and SPARQL querying are being consolidated into a common language and are thus independent from user interface design choices.

A lot of research have been done on overcoming specific challenges in regard to the *information overload* [26] [5], a phenomenon which occurs if the *information available exceeds the user's ability to process it.* Hence, in faceted exploration scenarios this happens when any of the involved sets, namely those of facets, facet values or result items, becomes too large. For example, datasets, such as DBpedia, may contain thousands of predicates, triggering the need for automatically determining an order of relevancy (possibly in regard to the current state). But also a single relevant facet may have several thousand of values (e.g. authors), hence clustering approaches, such as referred to in [19] can be applied to obtain reduction.

In [20] the authors summarize three approaches for presenting the user the most useful facets and facet values based on personalization. Thereby, these approaches fall into three categories: collaborative [15], content-based and ontology-based [24].

Several benchmarks for assessing the SPARQL performance of triple stores have been devised in the past, such as LUBM [11],

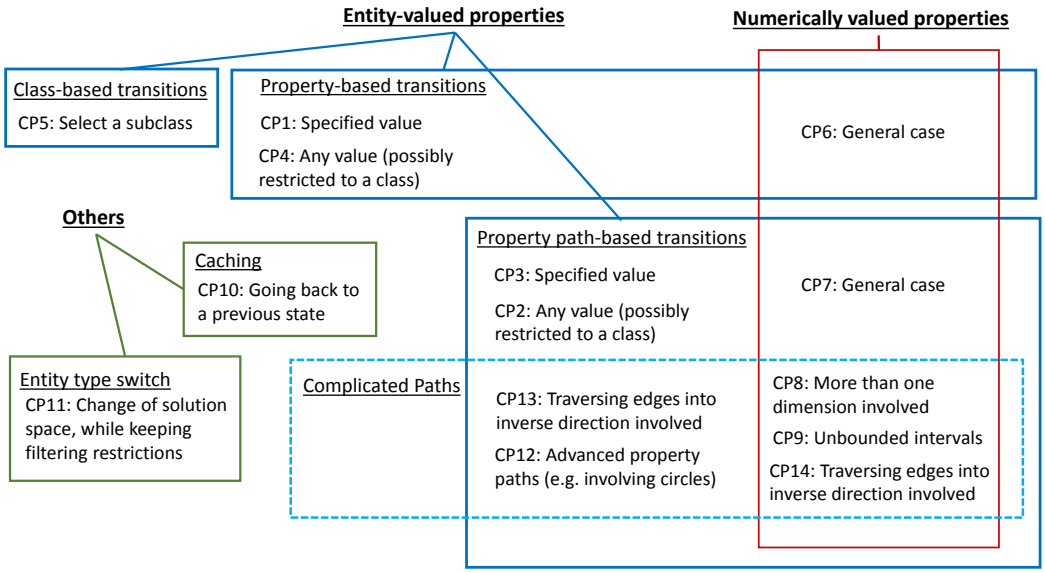

Figure 1: An overview of the choke points

SP2 [21], BSBM [6], and WatDiv [2]. Back in 2008, a lack for repeatable benchmarks for faceted browsing engines has been noted [15]. Since then, most progress in overcoming this lack has been made in the geo-spatial domain, where simulated faceted browsing scenarios have been applied to evaluate the performance of queries having varying numbers of restrictions along the spatial and thematic dimensions. These scenarios consist in general of simple facet selections and iterative panning and zooming of the map [9]. It is noteworthy, that [1][3] considers tile-based precomputed facet counts. However, to the best of our knowledge, our work introduces the first system dedicated to a detailed list of specific choke points of faceted browsing on RDF data.

3 CHOKE POINTS OF FACETED BROWSING

In a browsing scenario, where a dataset is being explored by applying and unapplying filter restrictions, the user expects that all transitions should run smoothly and within reasonable time. To enable efficient transitions, the information of the current state can be used instead of replying every one of the queries that make up the browsing scenario on basis of the entire dataset. Alternatively, strategically chosen precomputations may be taken into consideration when responding to a desired transition.

Therefore, the different choke points correspond each to certain transitions from one state to the other during the browsing scenario. For example, a transition might consist of choosing a property value of a directly related property (which we will call a property value based transition). Or, as another example, a property value behind a property path of length strictly larger than one edge may be chosen (which we will call a property path value based transition) . In the latter case it is, in comparison to the first example, a more challenging task to prepare the system for the upcoming transition, and it requires a more complex choice of the system's database structure.

In a survey on faceted browsing, Tzitzikas et al. list in [25] four basic faceted browsing choke points: class-based transitions, property-based transitions, property path-based transitions and entity type switches. For the evaluation of our benchmark we divided these points further. We split the property-based transitions according to Tzitzikas et al. into two cases. In the first case, one aims to fix the value behind the property (CP1), while in the second case one simply asks for the existence of any value realizing the property, possibly only restricted to lie in a certain class (CP4). If, in the latter case, containment in a specified class was requested, another choke point lies in a further restriction by selecting a subclass of the class in question (CP5). The same splitting into the two cases of selecting a specific value or merely asking for containment in some class is performed for property path value based transitions (CP3 & CP2).

Further, we consider properties with numerical values separately, for which the value can be restricted to a specified interval. Again, we distinguish behind numerical values behind a direct property (CP6) and behind a property path of length strictly greater than one (CP7). We also test whether multi-dimensional numerical data (CP8) or unbounded intervals (CP9) provide difficulties to the system. Caching abilities are checked by a simulation of a transition, where the user wants to recover a previous state (CP10).

The choke point of entity type switches (CP11) remains untouched from the formulation in [25], i.e., we simulate the transition of changing the solution space while keeping the current filter selections.

Finally, we distinguish between regular property paths and more complicated ones. We consider property-paths that involve traversing edges in the inverse direction for entity-valued properties (CP13) and numerical values (CP14) at the end of a property path.

[3]Latest version: https://github.com/GeoKnow/GeoBenchLab/tree/master/FacetBench

Additionally, we consider a class of advanced property paths (CP12). Under advanced property paths, we here understand property paths that involve an additional difficulty compared to the more simpler ones from CP2, CP3 and CP13. For example, this could be property paths in the knowledge graph that involve circles. In the dataset of train connections, described in Section 4.1, circles occur when a user is looking for all train stations that lie on a train route which also serves a certain different station.

Another example for a more advanced property path on the transport dataset is to select all stations that belong to one trip, which has a delay somewhere during its timeline. Here, the same result will appear multiple times in the result list and such double entries should be avoided.

Overall, this resulted in a list of 14 transitions that make up the choke points of our benchmark on faceted browsing. This means that systems can be compared on quite a large set of different aspects of performance. Note that the list is not exclusive in the sense that one transition may cover more than one of the mentioned choke points.

4 THE BENCHMARK

We implemented a benchmark on faceted browsing, which we present in this section. The benchmark is integrated into the HOBBIT platform[4], the description of which can be found in Deliverable 2.2.1 for the HOBBIT project[5]. On the HOBBIT platform, participants can readily test their systems after writing a system adapter following the Common API for the Mighty Storage Challenge (MOCHA)[6] of the ESWC 2017[7], which the benchmark of faceted browsing was part of. The source code of the benchmark is publicly available.[8]

In this section, we describe the different parts of our benchmark.

4.1 The underlying dataset

The data generator PoDiGG[9], described in [22], creates a transport dataset containing train connections between stations on an artificially created map. It also includes the possibility of delays that the trains may experience during their trajectory. The simulation of delays includes not only a time value, but also a reason for the delay (Figure 3).

For the integration of delays into the dataset, the Transport Disruption Ontology[10] [7] is used, which models possible events that can disrupt the schedule of travel or transport plans. A dataset from this generator is used for the development of simulated browsing sessions in our benchmark and its underlying ontology allows us to test on all choke points described in Section 3.

4.2 Scenarios

We created several lists of ordered SPAQRL queries, where each list simulates one browsing scenario (Figure 2). The development of the browsing scenarios took place with the aim to guarantee

Scenario 3

- Search space = Instances of connections

(1) Restrict the lat and long for stations 7
(2) Select a station. 3
(3) Select all connections that have a delay 2
(4) Select a delay reason. 4
(5) Deselect the delay reason. 10
(6) Restrict to delay value $> a$ 7,8,9
(7) Restrict to delay value $> b$ where $b < a$ 7,8,9
(8) Select a delay reason with subclasses 4
(9) Select a sub-reason 5
(10) Restrict to delay values $< c$ 7,8
(11) Restrict to delay values $< d$ where $d < c$ 7,8

Figure 2: One example scenario annotated with its related choke points

two requirements. Firstly, we wanted to come up with browsing sessions that make sense in a real-world browsing scenario. Secondly, the scenarios should also cover all types of transitions as specified by the choke points from Section 3. The overall workload of the benchmark comprises 173 SPARQL queries divided up into 11 scenarios, each simulating a single user browsing through the dataset. Every choke point appears at least a few times throughout all scenarios and one SPARQL query may contribute to the score related to several choke points.

To allow for randomness in the SPARQL queries that make up the benchmark, we wrote additional queries computing several variables, which are placeholders either for instances or for values of types xsd:dateTime, xsd:duration or xsd:decimal. In the case of instances, for each of the preparatory queries, one instance is randomly selected from the result list and plugged into the corresponding placeholder to complete one of the queries belonging to a browsing scenario. In the latter case, the value for the variable is either computed from dataset-dependent numbers (such as the minimum and maximum values for latitude and longitude value of stations), or selected from a strategically chosen interval in a random fashion. By choosing intervals, we can assure that values will definitely increase, or alternatively definitely decrease instead. Proceeding in this way of precomputing parameters, equips us with a method that can easily lead to varying browsing scenarios by simply changing a seed for the generation of random numbers. The seed can be specified when creating a benchmark on the platform by the user of the platform.

Having introduced randomness into the generation of our scenarios requires us to compute the gold standard on the fly. According to our tests, the open source triple store "Virtuoso"[11] correctly answered queries in a reliable fashion and therefore proved itself suitable as the system of choice to compute the gold standard. In a future version we aim to integrate a second system to compute

[4] http://master.project-hobbit.eu/
[5] https://project-hobbit.eu/wp-content/uploads/2017/04/D2.2.1.pdf.
[6] https://project-hobbit.eu/challenges/mighty-storage-challenge/
[7] http://2017.eswc-conferences.org/
[8] https://github.com/hobbit-project/faceted-benchmark
[9] https://github.com/PoDiGG/podigg-lc
[10] https://transportdisruption.github.io/
[11] https://virtuoso.openlinksw.com/

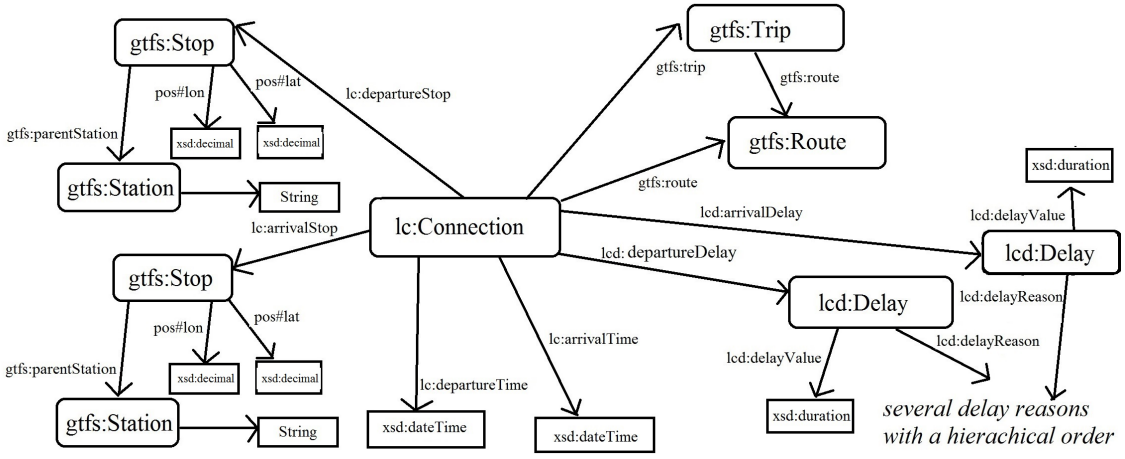

Figure 3: The ontology of the underlying dataset

results upfront and allow only result lists as a gold standard that both systems agree upon.

4.3 KPIs

As Key Performance Indicators, we set the usual suspects: precision, recall, F1-score and, of course, we collect the time between query formulation and receiving of an answer and record it in form of a score measuring queries per second. These four performance values (precision, recall, F1-score and query-per-second score) are registered over all queries of all scenarios combined and additionally for each of the above choke points individually. For the calculation in the latter case, the procedure is to only include those SPARQL queries that make up transitions corresponding to the choke point in question.

Measuring the KPIs of precision, recall and F1-score aims to check whether the returned list of results is incomplete (false negatives) or contains wrong results (false positives). Duplicated results here count as false positives, too. The idea behind measuring these quantities is that a system may choose to return incomplete results to save time, when the result sets become very large in size.

Whereas the just mentioned KPIs measure only the performance on instance retrievals (i.e. SELECT queries asking to return all instances satisfying or implementing certain relations) there is another type of queries important in faceted browsing: SELECT COUNT queries, which count the number of instances with respect to certain additional restrictions. These counts are important for suggesting possible transitions to the user and to guide in the browsing session. From a given state, only transitions leading to a sensible number of results should be suggested. Therefore, we included six SELECT COUNT queries into each of our 11 scenarios. Performance on these facet counts is also measured in more than one way to give a more complete picture of performance. In the following, we mean by 'error' on a single count query the absolute value of the differences between expected and received count result. For the performance on the count queries we record the following:

- The overall error, equal to the sum of individual errors over all count queries
- The average error which is simply the overall error divided by the number of queries.
- The overall error ratio as the overall error divided by the sum of expected count results
- The average error ratio as the sum of "error divided by expected result" over all queries.

5 EXPERIMENTS

The benchmark on faceted browsing is a result of the European funded project HOBBIT [18]. HOBBIT stands for "Holistic Benchmarking of big linked data" and aims to benchmark several tasks around the linked data value chain. As part of the HOBBIT project, experiments for our benchmark can be run on the HOBBIT platform. A detailed description on how to benchmark a system can be found on the project-related webpages[12].

In Figure 4, we see results from the MOCHA challenge at the ESWC 2017. In that challenge, three systems participated and one of them, Ontos system QUAD [13], experienced a time out during the run of the benchmark on faceted browsing. Unfortunately, we were therefore left with results for only two participating systems. Firstly, we have the implementation of a system adapter for Virtuoso 7.2 Open-Source Edition by OpenLink Software that follows a common API for all tasks of the MOCHA challenge. Because it follows the common API of the MOCHA challenge, this system served as our MOCHA baseline. The second contestant is the Virtuoso 8.0 Commercial Edition (beta release) by OpenLink Software. Our results show that both versions of Virtuoso show similar performance on instance retrieval and facet counts. More detailed results of that challenge will be published in the ESWC challenge proceedings [10]. Interestingly, the query-per-second score was the lowest for choke points 6, 7 and 12 (see Figure 4). While choke point 12 concerns

[12]https://github.com/hobbit-project/platform/wiki/Benchmark-your-system
[13]http://ontos.com/products/platform/

advanced property paths, both choke points 6 and 7 concern the performance on numerical data restrictions.

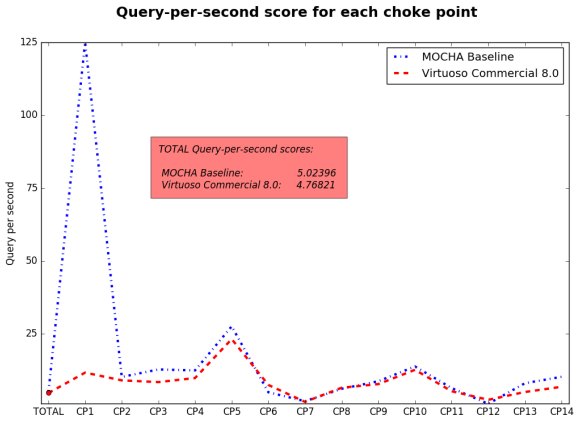

Figure 4: Preliminary results from the MOCHA challenge - Query-per-second score [10]

While the challenge was run with limited success in terms of the number of participating triple stores, further iterations of the faceted browsing challenge will be organized within the project over the next 18 months allowing further systems to participate.

6 ADDITIONAL PROBLEMS FOR GENERIC SOLUTIONS

In this section we will discuss additional choke points of faceted browsing. By example, we will demonstrate several difficulties in building a generic solution for faceted browsing of linked data functioning independently from the dataset and without human adjustments. These difficulties show a possible need for standards on how to model ontologies of datasets that allow to efficiently apply faceted browsing software.

We would like to stress that the examples we demonstrate here are not purely theoretical constructs, but situations we have encountered in industrial and other datasets.

In [23], the authors point out three challenges to extend faceted browsing solutions to large datasets: How to automatically generate metadata, how to decide on which facets to surface, and how to accurately preview facets and facet counts. These three points are related to the challenges discussed here, but they are of a different nature being rather general and less dependent on the modeling of the knowledge graph.

6.1 Example 1: Multiple facets, 'and' or 'or'

In the common graphical interface for faceted browsing, the facets to choose from are displayed in one of the corners. When a user wants to select one of these facets, he may set a check mark for the chosen facet to initiate the corresponding filtering equation. An obvious question is then: If a user sets multiple check marks, does that correspond to an 'and' or an 'or' relation? In other words, should both facet selections hold at the same time, or should only at least one of them hold?

Or should it even depend on the semantics? Suppose a user is looking to buy a car and searches for purchase opportunities with the help of a faceted browsing implementation. In this case, selecting multiple colors probably means that he prefers either one color or the other. But if the user is looking to buy a bouquet of flowers instead, then probably he wants a mixture of both colors, so both colors should be specified by a property.

This does not constitute a major hurdle as it is easy to image a more complex graphical interface that has one box for 'and' selections and another box for selections of type 'or'. This has for example been considered and implemented under the terminology of multi-facet search in Flamenco [12] and the Ontogator [14]. In multi-select faceting approaches[14], multiple facet selections within one facet are considered as a selection of type "or", while facet selections for different facets are considered as a selection of type "and". We still wanted to point out that if no consensus should be made whether to always mean 'and' or always mean 'or' when selecting multiple facets, then the graphical interface of a faceted browsing solution has to be at least as complex as to allow for both multiple facet selections of type 'and' and 'or'.

6.2 Example 2: Dependencies

Consider the excerpt from an industrial knowledge graph in Figure 5.

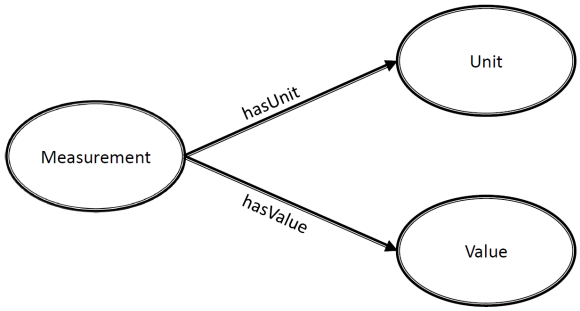

Figure 5: An excerpt from a knowledge graph demonstrating dependency issues

We have a class containing measurements of all sorts, each having a unit of measurement and a value. The problematic dependence in this simple example is the one between the value and the choice of the unit. A sensible restriction on the value only makes sense after the choice of the unit has taken place. Possibly even, the type of value (being a string, a boolean or a decimal number for example) of the measurement might depend on the chosen unit. A user-friendly interface should probably not suggest the numerical restriction of the value (with sensible maximal and minimal values) before the user has chosen the unit.

Probably a reader might want to argue that this issue is not a major issue either. Why not display both facets for unit and value at once and then the user should have the responsibility to choose values that make sense for the measurements he is looking for. But we should keep in mind that this is just an excerpt of a larger graph.

[14] http://yonik.com/multi-select-faceting/

The number of facets to choose from might be higher than the number of facets one can display at once in a legible fashion. In that case, how can we make sure that the facet specifying the value is not suggested to the user without the facet specifying the unit of measurement?

6.3 Example 3: Non-changing transitions

Let us consider now the excerpt from an industrial knowledge graph in Figure 6. We have a class of energy measurements with a property for the unit of measurements. In the dataset, the unit of every instance of the class of energy measurements is 'Joule'. While the declaration of the unit provides useful information to a user investigating the dataset, it is obsolete from a faceted browsing perspective.

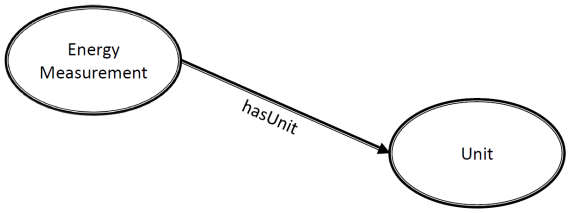

Figure 6: Some transitions may not change the solution space. Here, every energy measurement will have the unit 'Joule'.

This raises the following problems for the implementation. On the back end, the system should recognize that the transition cannot change the solution space and an immediate return is possible and hence would be desirable. On the front end, the transition of selecting the unit should not have a high priority to be displayed to the user.

A solution to this problem could be to agree upon that this situation should be modeled with an artificial triple containing metadata that specifies this situation, or even using the "ObjectAllValuesFrom" expression of the OWL Web Ontology Language[15].

In both cases the faceted browsing software can then pick up on the given information.

6.4 Example 4: Tree based transitions

While the above issues have rather been on the front end and concern which facets should be displayed and which should not, we consider now a choke point on the back end that does not fall under the 'traditional' choke points from Section 3. Consider the excerpt of an ontology in Figure 7. We have a class 'Measurement Group" which contains a collection of measurements at a certain point in time. The 'Measurement Group' contains several measurements, each having a unit and a value. We suppose that each instance of the 'Measurement Group' contains the same measurements of sensors, but taken at different times.

[15]https://www.w3.org/TR/owl2-direct-semantics

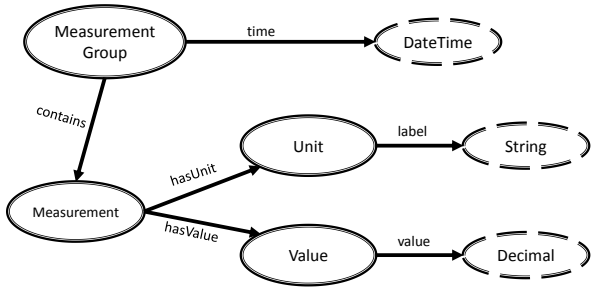

Figure 7: An excerpt from a knowledge graph demonstrating the necessity for tree based transitions

We suppose further that our search space is the the set of measurement groups. We may, for example, be interested in all incidences where a certain temperature had an exceptionally high value while a certain energy value was exceptionally low.

How would we proceed with the transitions considered in Section 3? First, we could make a property path value based transition, fixing the value 'Joule' behind the property path given by

$$"contains + hasUnit + label".$$

Note that this does not change the solution space, just as we discussed in 6.3, because every measurement group contains measurements of energy values. We could then try to additionally fix the temperature unit as 'Kelvin' at the end of the same property path as before. Also this does not change the solution space as in 6.3, and we assume the possibility to select multiple facets with the 'and' relation as discussed in 6.1. Now, trying to select a range for one of the values, energy or temperature, behind the property path given by

$$"contains + hasValue + value",$$

it is not clear which of the two measurements and units the value belongs to!

There are (at least) three ways out of this complication. Firstly, we could allow "tree-based transitions", where multiple paths and values are chosen simultaneously, and the involved paths span a tree. Since one can imagine almost arbitrarily more complex situations, supporting many such transitions might be a very hard problem for a generic solution.

Secondly, such situations could be modeled differently, for example by using the RDF Data Cube Vocabulary[16] or by providing metadata that follows a possible standard for faceted browsing.

Thirdly, human interaction could resolve the problems within the knowledge graph by extending it.

In any case, the bottom line remains that there are more difficulties in developing a generic faceted browsing system for RDF data than those covered by our choke points. An interesting study could be the investigation of possible RDF graph modeling standards for faceted browsing that limit the number of difficulties while being sufficiently expressive. Nevertheless, the choke points from Section 3 provide an important baseline of transitions that a solution with high performance should be efficient on.

[16]https://www.w3.org/TR/vocab-data-cube/

7 CONCLUSIONS

We demonstrated a transition centered benchmark on faceted browsing. Our benchmark provides the possibility for systems to test improvements on specific traditional faceted browsing choke points as the outcome of changes to their system's implementation.

The benchmark is based on browsing scenarios for a specific dataset consisting of RDF triples. In future work, it would be interesting to transfer the ideas to a benchmark, where both dataset and browsing scenarios are easily exchangeable. We further discussed problems of such an approach, by exemplifying that what is commonly considered as the choke points of faceted browsing is accompanied by more points of difficulty, when the goal is to develop generic software for faceted browsing that should be easily and automatically adjusted to new linked datasets. Our hope is that, by pointing out additional choke points, researchers will be motivated to overcome these problems next to the implementation of systems with good performance on the choke points from Section 3.

ACKNOWLEDGMENTS

This project has received funding from the European Union's H2020 research and innovation action program under grant agreement number 688227. The project runtime is December 2015 until November 2018.

REFERENCES

[1] 2010. LOD2 Deliverable 5.1.4 LOD2 GeoBench v2.0 Evaluation. http://svn.aksw.org/lod2/D5.1.4/public.pdf. (2010). Accessed: 2017-06-08.

[2] Güneş Aluç, Olaf Hartig, M. Tamer Özsu, and Khuzaima Daudjee. 2014. *Diversified Stress Testing of RDF Data Management Systems.* Springer International Publishing, Cham, 197–212. https://doi.org/10.1007/978-3-319-11964-9_13

[3] Kumaripaba Athukorala, Dorota Glowacka, Giulio Jacucci, Antti Oulasvirta, and Jilles Vreeken. 2016. Is exploratory search different? A comparison of information search behavior for exploratory and lookup tasks. *Journal of the Association for Information Science and Technology* 67, 11 (2016), 2635–2651. https://doi.org/10.1002/asi.23617

[4] Ori Ben-Yitzhak, Nadav Golbandi, Nadav Har'El, Ronny Lempel, Andreas Neumann, Shila Ofek-Koifman, Dafna Sheinwald, Eugene Shekita, Benjamin Sznajder, and Sivan Yogev. 2008. Beyond Basic Faceted Search. In *Proceedings of the 2008 International Conference on Web Search and Data Mining (WSDM '08)*. ACM, New York, NY, USA, 33–44. https://doi.org/10.1145/1341531.1341539

[5] Sonia Bergamaschi, Francesco Guerra, and Barry Leiba. 2010. Guest editors' introduction: information overload. *IEEE Internet Computing* 14, 6 (2010), 10–13. https://doi.org/doi.ieeecomputersociety.org/10.1109/MIC.2010.140

[6] Christian Bizer and Andreas Schultz. 2009. The Berlin SPARQL Benchmark. *Int. J. Semantic Web Inf. Syst.* 5, 2 (2009), 1–24. http://dblp.uni-trier.de/db/journals/ijswis/ijswis5.html#BizerS09

[7] David Corsar, Milan Markovic, Peter Edwards, and John D. Nelson. 2015. *The Transport Disruption Ontology.* Springer International Publishing, Cham, 329–336. https://doi.org/10.1007/978-3-319-25010-6_22

[8] Sébastien Ferré and Alice Hermann. 2012. Reconciling faceted search and query languages for the Semantic Web. *IJMSO* 7, 1 (2012), 37–54. https://doi.org/10.1504/IJMSO.2012.048508

[9] George Garbis, Kostis Kyzirakos, and Manolis Koubarakis. 2013. *Geographica: A Benchmark for Geospatial RDF Stores (Long Version).* Springer Berlin Heidelberg, Berlin, Heidelberg, 343–359. https://doi.org/10.1007/978-3-642-41338-4_22

[10] Kleanthi Georgala, Mirko Spasic, Milos Jovanovik, Henning Petzka, Michael Röder, and Axel-Cyrille Ngonga Ngomo. MOCHA2017: The Mighty Storage Challenge at ESWC 2017. In *Accepted for publication in the ESWC 2017 Challenge Proceedings, Portoroz, Slovenia, 2017.* Springer International Publishing.

[11] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. 2005. LUBM: A benchmark for OWL knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web* 3, 2 (2005), 158 – 182. https://doi.org/10.1016/j.websem.2005.06.005 Selcted Papers from the International Semantic Web Conference, 2004.

[12] Marti Hearst, Ame Elliott, Jennifer English, Rashmi Sinha, Kirsten Swearingen, and Ka-Ping Yee. 2002. Finding the Flow in Web Site Search. *Commun. ACM* 45, 9 (Sept. 2002), 42–49. https://doi.org/10.1145/567498.567525

[13] Jiří Helmich, Jakub Klímek, and Martin Nečaský. 2014. *Visualizing RDF Data Cubes Using the Linked Data Visualization Model.* Springer International Publishing, Cham, 368–373. https://doi.org/10.1007/978-3-319-11955-7_50

[14] Eero Hyvönen, Samppa Saarela, and Kim Viljanen. 2004. *Application of Ontology Techniques to View-Based Semantic Search and Browsing.* Springer Berlin Heidelberg, Berlin, Heidelberg, 92–106. https://doi.org/10.1007/978-3-540-25956-5_7

[15] Jonathan Koren, Yi Zhang, and Xue Liu. 2008. Personalized Interactive Faceted Search. In *Proceedings of the 17th International Conference on World Wide Web (WWW '08)*. ACM, New York, NY, USA, 477–486. https://doi.org/10.1145/1367497.1367562

[16] Gary Marchionini. 2006. Exploratory Search: From Finding to Understanding. *Commun. ACM* 49, 4 (April 2006), 41–46. https://doi.org/10.1145/1121949.1121979

[17] Michael Martin, Konrad Abicht, Claus Stadler, Axel-Cyrille Ngonga Ngomo, Tommaso Soru, and Sören Auer. 2015. Cubeviz: Exploration and visualization of statistical linked data. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 219–222.

[18] Axel-Cyrille Ngonga Ngomo, Alejandra García-Rojas, and Irini Fundulaki. 2016. HOBBIT: Holistic Benchmarking of Big Linked Data. *ERCIM News* 2016, 105 (2016). http://ercim-news.ercim.eu/en105/r-i/hobbit-holistic-benchmarking-of-big-linked-data

[19] Eyal Oren, Renaud Delbru, and Stefan Decker. 2006. *Extending Faceted Navigation for RDF Data.* Springer Berlin Heidelberg, Berlin, Heidelberg, 559–572. https://doi.org/10.1007/11926078_40

[20] Giovanni Maria Sacco and Yannis Tzitzikas. 2009. *Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience* (1st ed.). Springer Publishing Company, Incorporated.

[21] Michael Schmidt, Thomas Hornung, Michael Meier, Christoph Pinkel, and Georg Lausen. 2010. *SP2Bench: A SPARQL Performance Benchmark.* Springer Berlin Heidelberg, Berlin, Heidelberg, 371–393. https://doi.org/10.1007/978-3-642-04329-1_16

[22] Ruben Taelman, Ruben Verborgh, Tom De Nies, and Erik Mannens. 2017. PoDiGG: A Public Transport RDF Dataset Generator. In *Proceedings of the 26th International Conference Companion on World Wide Web.* http://rubensworks.net/raw/publications/2017/PodiggPublicTransportRdfDatasetGenerator.pdf

[23] Jaime Teevan, Susan Dumais, and Zachary Gutt. 2008. Challenges for Supporting Faceted Search in Large, Heterogeneous Corpora like the Web. https://www.microsoft.com/en-us/research/publication/challenges-supporting-faceted-search-large-heterogeneous-corpora-like-web/

[24] Michal Tvarozek and Mária Bieliková. 2007. Personalized faceted navigation for multimedia collections. In *Semantic Media Adaptation and Personalization, Second International Workshop on.* IEEE, 104–109.

[25] Yannis Tzitzikas, Nikos Manolis, and Panagiotis Papadakos. 2017. Faceted exploration of RDF/S datasets: a survey. *Journal of Intelligent Information Systems* 48, 2 (April 2017), 329–364. https://doi.org/10.1007/s10844-016-0413-8

[26] Bifan Wei, Jun Liu, Qinghua Zheng, Wei Zhang, Xiaoyu Fu, and Boqin Feng. 2013. A Survey of Faceted Search. *J. Web Eng.* 12, 1-2 (Feb. 2013), 41–64. http://dl.acm.org/citation.cfm?id=2481562.2481564